WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Term	Documents
VIRTUAL	59102
VIRTUALS	3
WAN	13416
WANS	1763
LAN	28025
LANS	6585
(VIRTUAL ADJ1 (WAN OR LAN)).TI..USPT.	15
((VIRTUAL ADJ1 (WAN OR LAN)).TI.).USPT.	15

Database:
US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L3

Refine Search

Recall Text | Clear

Search History

DATE: Wednesday, December 03, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
	DB=USPT; PLUR=YES; OP=ADJ		
L3	(virtual adj1 (wan or lan)).ti.	15	L3
L2	(virtual adj1 (wan or lan))	362	L2
L1	(virtual with (wan or lan))	980	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 10 of 15 returned.

☐ 1. Document ID: US 6578021 B1

L3: Entry 1 of 15 File: USPT Jun 10, 2003

DOCUMENT-IDENTIFIER: US 6578021 B1
TITLE: Method and system for classifying network devices in virtual LANs

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

---

☐ 2. Document ID: US 6570875 B1

L3: Entry 2 of 15 File: USPT May 27, 2003

DOCUMENT-IDENTIFIER: US 6570875 B1
TITLE: Automatic filtering and creation of virtual LANs among a plurality of switch ports

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

---

☐ 3. Document ID: US 6560236 B1

L3: Entry 3 of 15 File: USPT May 6, 2003

DOCUMENT-IDENTIFIER: US 6560236 B1
TITLE: Virtual LANs

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

---

☐ 4. Document ID: US 6473803 B1

L3: Entry 4 of 15 File: USPT Oct 29, 2002

DOCUMENT-IDENTIFIER: US 6473803 B1
TITLE: Virtual LAN interface for high-speed communications between heterogeneous computer systems

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 5. Document ID: US 6356551 B1

L3: Entry 5 of 15 File: USPT Mar 12, 2002

DOCUMENT-IDENTIFIER: US 6356551 B1
TITLE: Method and network switch having dual forwarding models with a virtual lan overlay

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 6. Document ID: US 6269098 B1

L3: Entry 6 of 15 File: USPT Jul 31, 2001

DOCUMENT-IDENTIFIER: US 6269098 B1
TITLE: Method and apparatus for scaling number of virtual lans in a switch using an indexing scheme

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 7. Document ID: US 6269076 B1

L3: Entry 7 of 15 File: USPT Jul 31, 2001

DOCUMENT-IDENTIFIER: US 6269076 B1
TITLE: Method of resolving split virtual LANs utilizing a network management system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

☐ 8. Document ID: US 6085238 A

L3: Entry 8 of 15 File: USPT Jul 4, 2000

DOCUMENT-IDENTIFIER: US 6085238 A
**** See image for Certificate of Correction ****
TITLE: Virtual LAN system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

☐ 9. Document ID: US 6061334 A

L3: Entry 9 of 15 File: USPT May 9, 2000

DOCUMENT-IDENTIFIER: US 6061334 A
TITLE: Apparatus and method for assigning virtual LANs to a switched network

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

☐ 10. Document ID: US 6038608 A

L3: Entry 10 of 15 File: USPT Mar 14, 2000

DOCUMENT-IDENTIFIER: US 6038608 A
**** See image for Certificate of Correction ****
TITLE: Virtual LAN system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

Generate Collection | Print

Term	Documents
VIRTUAL	59102
VIRTUALS	3
WAN	13416
WANS	1763
LAN	28025
LANS	6585
(VIRTUAL ADJ1 (WAN OR LAN)).TI..USPT.	15
((VIRTUAL ADJ1 (WAN OR LAN)).TI.).USPT.	15

Display Format: KWIC | Change Format

Previous Page Next Page

WEST

Generate Collection

L7: Entry 7 of 26 File: USPT Dec 5, 2000

DOCUMENT-IDENTIFIER: US 6157647 A
TITLE: Direct addressing between VLAN subnets

Detailed Description Text (2):
Referring to FIG. 1, a local area network ("LAN") 10 includes a plurality of virtual LANs ("VLANs") 12, 14, 16, 18 which are interconnected along a path 20 by routers 22, 24, 26. Each VLAN includes a plurality of computer devices 28, 30, 32, 34, 36, 38, 40, 42, 44, 46 (sometimes referred to as "hosts"), such as workstations. The VLANs are connected to computer networks 48, 50, 52 outside of the LAN 10 through the routers 22, 24, 26, respectively, which serve as gateways.

WEST

Generate Collection

L7: Entry 8 of 26 File: USPT Aug 8, 2000

DOCUMENT-IDENTIFIER: US 6101188 A
TITLE: Internetworking router

Brief Summary Text (8):
That is, some of conventionally used network protocols have a mechanism which presupposes a local address and automatically rewrites an interface address not into a global MAC address but into a unique local address to be used. A router which relays a network of the type just described assigns a same local MAC address to two or more interfaces. Accordingly, if the router is connected between virtual LANs, such a situation that the same MAC address is connected in a plurality of virtual LANs occurs. In such a construction, a stable relaying operation cannot be expected.

Brief Summary Text (11):
The second problem of the prior art resides in that, if a switching hub constructing virtual LANs has a traffic of the third layer which spans a plurality of virtual LANs, since relaying is impossible with just a bridge function, an external router of the third layer is required.

Brief Summary Text (16):
In order to attain the object described above, according to the present invention, there is provided an internetworking router for a LAN switching hub apparatus which includes an address learning table which holds as a set for a certain period of time a source second layer address for a packet received from a plurality of LAN interfaces and the LAN interface of the plurality of LAN interfaces which has received the packet, wherein a destination second layer address is extracted from the received packet and the address learning table is searched using the destination second layer address as a key to specify a LAN interface which corresponds to the destination second layer address from within the plurality of LAN interfaces, and then the packet is relayed and sent out only to the specified LAN interface, constructed such that the internetworking router has a plurality of bridge groups which construct a plurality of virtual LANs by arbitrarily combining those of the plurality of LAN interfaces

Brief Summary Text (19):
The internetworking router of the present invention described above may be constructed such that the internetworking router includes a plurality of logical interfaces including virtual LAN interfaces and routing processing means for relaying the plurality of logical interfaces, and, when the packet is to be relayed between a first bridge group and a second bridge group from among the plurality of bridge groups, a third logical interface for relaying the first bridge group and the second bridge group from among the plurality of logical interfaces delivers address pointer information to a buffer into which the packet has been stored using a port ID of the virtual LAN interface between the bridge processing means and the routing processing means.

Brief Summary Text (20):
When the third logical interface from among the plurality of logical interfaces is to deliver the address pointer information to the buffer into which the packet has been stored using the port ID of the virtual LAN interface between the bridge processing means and the routing processing means, the address pointer information may be delivered in a multiplexed state into a reception buffer queue or a transmission buffer queue.

CLAIMS:

1. An internetworking router for a LAN switching hub apparatus comprising:

an address learning table which holds as a set for a certain period of time a source second layer address for a packet received from a plurality of LAN interfaces and the LAN interface of said plurality of LAN interfaces which has received said packet;

a plurality of bridge groups which construct a plurality of virtual LANs by arbitrarily combining those of said plurality of LAN interfaces which can be relayed to each other;

said plurality of bridge groups having said address learning tables for relaying process independently of each other, and

bridge processing means which performs learning processing the address learning table of said plurality of bridge groups to which the received packet belongs to;

wherein a destination second layer address is extracted from said received packet and said address learning table is searched using said destination second layer address as a key to specify a LAN interface which corresponds to said destination second layer address from within said plurality of LAN interfaces, and then said packet is relayed and sent out only to said specified LAN interface.

2. An internetworking router according to claim 1, further comprising

a plurality of logical interfaces including virtual LAN interfaces and routing processing means for relaying said plurality of logical interfaces;

wherein, when said packet is to be relayed between a first bridge group and a second bridge group from among said plurality of bridge groups, a third logical interface for relaying said first bridge group and said second bridge group from among said plurality of logical interfaces delivers address pointer information to a buffer into which said packet has been stored using a port ID of said virtual LAN interface between said bridge processing means and said routing processing means.

3. An internetworking router according to claim 2,

wherein, when said third logical interface from among said plurality of logical interfaces is to deliver the address pointer information to the buffer into which said packet has been stored using the port ID of said virtual LAN interface between said bridge processing means and said routing processing means, said address pointer information is delivered in a multiplexed state into a reception buffer queue or a transmission buffer queue.

WEST

Generate Collection

L7: Entry 11 of 26 File: USPT Apr 4, 2000

DOCUMENT-IDENTIFIER: US 6047320 A
TITLE: Network managing method and system

Detailed Description Text (12):
FIG. 4 is a diagram illustrating a network managing form provided by the network managing system of this embodiment. In FIG. 4, a network to be managed is divided into and separately defined as a virtual LAN space/network space 2000 and a virtual service space 3000. The virtual LAN space/network space 2000 corresponds to any or a combination of the Virtual LAN (by Port) layer 2100, the Virtual LAN (by MAC) layer 2200, and the Virtual Network (by IP) layer 2300 in FIG. 2. The virtual service space 3000 and the Physical Network space 1000 correspond to the Virtual Service layer 3000 and the Physical Network layer 1000 in FIG. 2, respectively. In the virtual LAN/network space 2000, a plurality of virtual networks 2310, 2320, 2330, 2340, 2350, having closed area characteristics, and their connection states are defined. A path 2101 in the figure indicates that the virtual networks 2310, 2320 are connected to each other such that they can communicate through the path 2101. Here, the configurations of nodes 2210, 2220, 2230, 2240 in each virtual network are managed by a system administrator 20, while the connection between the virtual networks is managed by a network administrator 10. In addition, the virtual service space 3000 is managed by a service administrator 30.

WEST

Generate Collection

L2: Entry 1 of 1 File: USPT Mar 14, 2000

DOCUMENT-IDENTIFIER: US 6038608 A
**** See image for Certificate of Correction ****
TITLE: Virtual LAN system

Abstract Text (1):
By providing a plurality of virtual LAN's such that ports communicating according to one protocol are grouped and communication is performed between the ports in the group, a communication is possible according to a plurality of protocols and it is possible to enter into other virtual LAN's than a virtual LAN to which a connection is made.

Brief Summary Text (3):
The present invention relates to a virtual LAN system and, particularly, to a virtual LAN system in which a virtual LAN is constructed every protocol.

Brief Summary Text (5):
In order to construct a plurality of virtual LAN's every protocol indicative of communication procedures in a virtual LAN system of this kind, it has been usual, as disclosed in Japanese Patent Application Laid-open No. Sho 64-54954, to determine a protocol higher in level than a network layer of OSI reference model used in a communication every LAN (referred to as "upper protocol", hereinafter) such that a communication is possible in a LAN according to only the determined protocol.

Brief Summary Text (6):
The term "virtual LAN" in this description means a LAN which services a host terminal by connecting the host terminal to the LAN through not a physical port of the host terminal but a logical connection.

Brief Summary Text (7):
Further, in order to enable the host terminal to belong to a plurality of virtual LAN's, virtual LAN's to which the host terminal can belong are preliminarily registered in a virtual server for controlling the construction of the virtual LAN's and a connection is made to the virtual LAN server by assigning the virtual LAN to be used in a communication when the host terminal starts the communication.

Brief Summary Text (8):
Since such conventional virtual LAN system determines the upper protocol to be used in a communication every LAN in order to construct a plurality of virtual LAN's every protocol indicative of the communication procedures, a communication within the LAN must be performed by using only this protocol. Therefore, there is a problem that the host terminal can not perform communication by using a plurality of upper protocols within the connected LAN. Further, since, in order to enable the host terminal to belong to a plurality of virtual LAN's, virtual LAN's to which the host terminal can belong are preliminarily registered in a virtual server for controlling the construction of the virtual LAN's and a connection is made to the virtual LAN server by assigning the virtual LAN to be used in a communication when the host terminal starts the communication, there is another problem that can not use other virtual LAN's than the virtual LAN to which the connection is made.

Brief Summary Text (10):
An object of the present invention is to provide a virtual LAN system in which a host terminal can perform communication by using a plurality of upper protocols within a LAN to which the host terminal is connected and can use other virtual LAN's than a virtual LAN to which a connection is made.

Brief Summary Text (11):
In the virtual LAN system according to the present invention, a plurality of virtual LAN's are provided in each of which a communication is performed between ports in a port group including a plurality of ports which communicate with each other by using the same protocol.

Brief Summary Text (12):
Further, the virtual LAN system according to the present invention which uses an intelligent switching hub device including a plurality of ports and a frame switch for outputting a frame to one of the ports according to a destination MAC (Media Access Control) address indicative of the one port as a destination and contained in the frame received by one of the ports, is featured by that the frame switch comprises a port table having pointers A indicative of next tables corresponding to the respective ports, a plurality of protocol ID tables produced correspondingly to the pointers A of the respective port tables and having pointers B indicative of respective next tables corresponding in number to protocols which can be communicated with the ports corresponding to the respective pointers A, a plurality of virtual LAN data tables produced for respective virtual LAN's preliminarily determined in the frame switch and having pointers C indicative of respective next tables appointed by the pointers B in the protocol ID tables and a port information indicative of the ports belonging to the respective virtual LAN's and a plurality of MAC forwarding tables produced correspondingly to the respective pointers C in the virtual LAN data tables and having port numbers of the ports corresponding to the respective pointers A and produced correspondingly to transmitting MAC addresses contained in the frame received by the ports corresponding to the pointers A in the port tables and indicative of transmitters and a count value of a counter for continuously counting from a time at which the port numbers are registered.

Brief Summary Text (13):
Further, the virtual LAN system according to the present invention which uses an intelligent switching hub device including a plurality of ports and a frame switch for outputting a frame to one of the ports according to a destination MAC address indicative of the one port as a destination and contained in the frame received by one of the ports, is featured by that the frame switch receives the frame at one of the ports, searches the pointer A from the port table according to the number of the port which received the frame, searches the pointer B from the protocol ID table indicated by the pointer A according to a protocol ID indicative of the kind of protocol contained in the received frame and used in the frame communication, extracts the pointer C from the virtual LAN data table indicated by the pointer B, searches the MAC forwarding table indicated by the pointer C according to the transmitting MAC address contained in the received frame and, when there is a constructive component of the MAC forwarding table corresponding to the transmitting MAC address, sets the port number of the received frame to the port number of the constructive component and resets the count value of the timer of the constructive component, newly registers the port number in the MAC forwarding table correspondingly to the transmitting MAC address when there is no constructive component, searches the MAC forwarding table indicated by the pointer C using the destination MAC address contained in the frame and, when there is a constructive component of the MAC forwarding table corresponding to the destination MAC address, sends the received frame to the port indicated by the port number of the searched constructive component, and sends the received frame to all of the ports indicated by the port numbers which can communicate by the virtual LAN indicated by the port information in the virtual LAN table having the pointers C indicated by the MAC forwarding table when there is no constructive component.

Drawing Description Text (2):
FIG. 1 is a block diagram showing an embodiment of a virtual LAN system according to the present invention;

Drawing Description Text (4):
FIG. 3 shows a port information indicative of a port belonging to a virtual LAN; and

Detailed Description Text (3):
FIG. 1 is a block diagram showing an embodiment of a virtual LAN system according to

the present invention. In this embodiment, the virtual LAN system, which uses an intelligent switching hub device 1 comprising a plurality of ports 3 which include physical ports which can be physically connected or logical ports which can be logically connected and a frame switch 2 for outputting a frame 9 to one of the ports 3 according to a destination MAC (Media Access Control) address indicative of the one port as a destination and contained in the frame 9 received by one of the ports 3, is featured by that the frame switch 2 comprises a port table 4 having pointers A indicative of next tables corresponding to the respective ports 3, a plurality of protocol ID tables 5 produced correspondingly to the respective pointers A and having pointers B indicative of respective next tables, a plurality of virtual LAN data tables 6 produced for respective virtual LAN's preliminarily determined in the frame switch 2 and having pointers C indicative of respective next tables and a port information 8 indicative of the ports 3 belonging to the respective virtual LAN's and a plurality of forwarding tables 7 produced correspondingly to the respective pointers C in the virtual LAN data tables 6 and having port numbers of the ports 3 corresponding to the respective pointers A and a count value of a counter for continuously counting from a time at which the port numbers are registered. Incidentally, the MAC is a control using a MAC sub-layer among data link layers of a hierarchical model of LAN.

Detailed Description Text (4):
An operation of the virtual LAN system according to this embodiment will be described in detail with reference to FIGS. 2, 3 and 4.

Detailed Description Text (5):
FIG. 2 shows a construction of the frame switch. The frame switch 2 comprises the port table 4 having pointers A indicative of next tables corresponding to the respective ports 3, the plurality of the protocol ID tables 5 produced correspondingly to the respective pointers A and having pointers B indicative of respective next tables the number of which corresponds to the number of the protocols according to which the ports 3 corresponding to the pointers A can communicate, the plurality of the virtual LAN data tables 6 produced for respective virtual LAN's preliminarily determined in the frame switch 2 and having the pointers C indicative of respective next tables and the port information 8 indicative of the ports 3 belonging to the respective virtual LAN's and the plurality of the forwarding tables 7 produced correspondingly to the respective pointers C in the virtual LAN data tables 6 and having the port numbers of the ports 3 corresponding to the respective pointers A and the count value of the timer for continuously counting from the time at which the port numbers are registered.

Detailed Description Text (6):
FIG. 3 shows an example of the port information indicating a port belonging to the virtual LAN. The shown port information indicates that the ports P2, P3, P5 and Pn of the n ports 3 belong to the virtual LAN.

Detailed Description Text (8):
In FIG. 1, the frame switch 2 of the intelligent switching hub device 1 receives the frame 9 at one of its ports 3 and searches the pointer A from the port table 4 according to the number of the port which received the frame 9 as shown in FIG. 2. According to the protocol ID indicative of the kind of the protocol contained in the received frame 9, used in the communication of the frame 9 and shown in FIG. 4, the pointer B is searched from the protocol ID table 5 indicated by the pointer A, as shown in FIG. 2. Then, as shown in FIG. 2, the pointer C is extracted from the virtual LAN data table 6 indicated by the pointer B and the MAC forwarding table 7 indicated by the pointer C with using the transmitting MAC address contained in the received frame 9 and shown in FIG. 4. When there is the constructive component of the MAC forwarding table 7 corresponding to the transmitting MAC address, the number of the port which received the frame 9 is set to the port number of the searched constructive component and the count value of the timer of this constructive component is reset. When there is no constructive component, the number of the port 3 is newly registered in the MAC forwarding table 7 correspondingly to the transmitting MAC address as the port number. In this case, the port number and the count value of the timer corresponding to this port number and continuing the counting from the time at which the port number is registered become the constructive components of the MAC forwarding table 7. Then, the MAC forwarding

table 7 indicated by the pointer C is searched by using the destination MAC address contained in the frame 9 and shown in FIG. 4 and, when there are constructive components of the MAC forwarding table 7 corresponding to the destination MAC address, the frame 9 received at the port 3 having the port number of the searched constructive components is sent. When there is no constructive component, the received frame 9 is sent to all of the ports 3 which are indicated in the port information 8 shown in FIG. 3 and contained in the virtual LAN data table 6 having the pointer C indicating the MAC forwarding table 7 and are enabled to communicate by the virtual LAN. When the count value of the timer among other constructive components produced in the MAC forwarding table 7 correspondingly to the transmitting MAC address contained in the frame 9 received by the port 3 and indicative of the transmitter exceeds a predetermined value, the constructive component corresponding to the transmitting MAC address is deleted from the MAC forwarding table 7.

Detailed Description Text (9):
As described hereinbefore, according to the virtual LAN system of the present invention, a protocol ID table is provided every part for defining a protocol according to which the port can communicate. Therefore, a communication is possible according to a plurality of upper protocols by using a port.

Detailed Description Text (10):
Further, since a virtual LAN data table is provided which makes a virtual LAN to which the frame belongs passible to define by a protocol ID contained in a received frame, it is possible to join other virtual LAN's than a connection is provided.

Detailed Description Text (11):
Further, since, in a case where a destination MAC address in the received frame is not learnt (indicating that there is no constructive component corresponding to this MAC address registered in the MAC forwarding table), the frame is transmitted to only ports which can communicate through a virtual LAN corresponding to a protocol ID which is indicated in the port information of the virtual LAN data table and corresponds to the protocol ID in this frame. Therefore, there are only frames in the virtual LAN, which are to be communicated with according to a protocol corresponding to the virtual LAN, so that an intra-LAN communication can be performed with high transmission efficiency without unnecessary load on the virtual LAN.

CLAIMS:

1. A virtual LAN system comprising a plurality of virtual LANs and an intelligent switching hub device with a plurality of ports for conducting protocol-based communications, wherein ports communicating according to one protocol are grouped and communication is performed between said ports in said group, wherein said virtual LAN system includes a frame switch for outputting a frame to one of said ports according to a destination MAC address indicative of said one port as a destination and contained in said frame received by said one of said ports, and wherein said frame switch comprises:

a port table having pointers A indicative of next tables corresponding to said respective ports;

a plurality of protocol ID tables produced correspondingly to said pointers A of said respective port tables and having pointers B indicative of respective next tables corresponding in number to protocols which can be communicated with said ports corresponding to said respective pointers A;

a plurality of virtual LAN data tables produced for respective virtual LAN preliminarily determined in said frame switch and having pointers C indicative of respective next tables appointed by said pointers B in said protocol ID tables and a port information indicative of said ports belonging to said respective virtual LANs; and

a plurality of MAC forwarding tables produced correspondingly to said respective pointers C in said virtual LAN data tables and having port numbers of said ports

corresponding to said respective pointers A and produced corresponding to transmitting MAC addresses contained in said frame received by said ports corresponding to said pointers A in said port tables and indicative of transmitters and a count value of a counter for continuously counting from a time at which said port numbers are registered.

2. A virtual LAN system as claimed in claim 1, wherein said virtual LAN system uses an intelligent switching hub device including a plurality of ports and a frame switch for outputting a frame to one of said ports according to a destination MAC address indicative of said one port as a destination and contained in said frame received by said one of said ports and wherein said frame switch receives said frame at one of said ports, searches said pointer A from said port table according to the number of said port which received said frame, searches said pointer B from said protocol ID table indicated by said pointer A according to a protocol ID indicative of the kind of protocol contained in said received frame and used in said frame communication, extracts said pointer C from said virtual LAN data table indicated by said pointer B, searches said MAC forwarding table indicated by said pointer C according to said transmitting MAC address contained in said received frame and, when there is a constructive component of said MAC forwarding table corresponding to said transmitting MAC address, sets said port number of said received frame to said port number of said constructive component and resets the count value of said timer of said constructive component, newly registers said port number in said MAC forwarding table correspondingly to said transmitting MAC address when there is no constructive component, searches said MAC forwarding table indicated by said pointer C using said destination MAC address contained in said frame and, when there is a constructive component of said MAC forwarding table corresponding to said destination MAC address, sends said received frame to said port indicated by said port number of said searched constructive component, and sends said received frame to all of said ports indicated by said port numbers which can communicate by said virtual LAN indicated by the port information in said virtual LAN table having said pointers C indicated by said MAC forwarding table when there is no constructive component.

3. A virtual LAN system as claimed in claim 1, wherein said frame switch removes the constructive component corresponding to said MAC address from said MAC forwarding table, when the count value of said timer of the constructive components of said MAC forwarding table produced correspondingly to said respective transmitting MAC addresses indicative of a transmitter and contained in said frame received by said port exceeds a predetermined value.

4. A virtual LAN system as claimed in claim 3 wherein said virtual LAN system comprises an intelligent switching hub device including a plurality of ports and a frame switch for outputting a frame to one of said ports according to a destination MAC address indicative of said one port as a destination and contained in said frame received by said one of said ports and wherein said frame switch receives said frame at one of said ports, searches said pointer A from said port table according to the number of said port which received said frame, searches said pointer B from said protocol ID table indicated by said pointer A according to a protocol ID indicative of the kind of protocol contained in said received frame and used in said frame communication, extracts said pointer C from said virtual LAN data table indicated by said pointer B, searches said MAC forwarding table indicated by said pointer C according to said transmitting MAC address contained in said received frame and, when there is a constructive component of said MAC forwarding table corresponding to said transmitting MAC address, sets said port number of said received frame to said port number of said constructive component and resets the count value of said timer of said constructive component, newly registers said port number in said MAC forwarding table correspondingly to said transmitting MAC address when there is no constructive component, searches said MAC forwarding table indicated by said pointer C using said destination MAC address contained in said frame and, when there is a constructive component of said MAC forwarding table corresponding to said destination MAC address, sends said received frame to said port indicated by said port number of said searched constructive component, and sends said received frame to all of said ports indicated by said port numbers which can communicate by said virtual LAN indicated by the port information in said virtual LAN table having said pointers C indicated by said MAC forwarding table when there is no constructive

component.

5. A virtual LAN system as claimed in claim 1 wherein said ports include physical ports which can be connected physically and logic ports which can be connected logically.

6. A virtual LAN system as claimed in claim 2, wherein said frame switch removes the constructive component corresponding to said MAC address from said MAC forwarding table when the count value of said timer of the constructive components of said MAC forwarding table produced corresponding to said respective transmitting MAC addresses indicative of a transmitter and contained in said frame received by said port exceeds a predetermined value.

7. A virtual LAN system as claimed in claim 2, wherein said ports include physical ports which can be connected physically and logical ports which can be connected logically.

8. A virtual LAN system comprising:

a plurality of virtual LANs;

an intelligent switching hub device with a plurality of ports for conducting protocol-based communications, wherein ports communicating according to one protocol are grouped and communication is performed between said ports in said group, and wherein a virtual LAN is constructed with every protocol according to which said respective ports can perform a communication; and

a frame switch for outputting a frame to one of said ports according to a destination MAC address indicative of said one port as a destination and contained in said frame received by said one of said ports, and wherein said frame switch comprises:

a port table having pointers A indicative of next tables corresponding to said respective ports;

a plurality of protocol ID tables produced correspondingly to said pointers A of said respective port tables and having pointers B indicative of respective next tables corresponding in number to protocols which can be communicated with said ports corresponding to said respective pointers A;

a plurality of virtual LAN data tables produced for respective virtual LAN preliminarily determined in said frame switch and having pointers C indicative of respective next tables appointed by said pointers B in said protocol ID tables and a port information indicative of said ports belonging to said respective virtual LANs; and

a plurality of MAC forwarding tables produced correspondingly to said respective pointers C in said virtual LAN data tables and having port numbers of said ports corresponding to said respective pointers A and produced corresponding to transmitting MAC addresses contained in said frame received by said ports corresponding to said pointers A in said port tables and indicative of transmitters and a count value of a counter for continuously counting from a time at which said port numbers are registered.

9. The virtual LAN system of claim 3, wherein said ports include physical ports which can be connected physically and logic ports which can be connected logically.

First Hit Fwd Refs

Tunnel

Generate Collection

L14: Entry 8 of 12 File: USPT Jun 20, 2000

DOCUMENT-IDENTIFIER: US 6078957 A
TITLE: Method and apparatus for a TCP/IP load balancing and failover process in an internet protocol (IP) network clustering system

Brief Summary Text (5):
A number of companies have recently announced current or proposed VPN products and/or systems which variously support IPSec, IKE (ISAKMP/Oakley) encryption-key management, as well as draft protocols for Point-to-Point Tunneling protocol (PPTP), and Layer 2 Tunneling protocol (L2TP) in order to provide secure traffic to users. Some of these products include IBM's Nways Multiprotocol Routing Services.TM. 2.2, Bay Networks Optivity.TM. and Centillion.TM. products, Ascend Communication's MultiVPN.TM. package, Digital Equipment's ADI VPN product family, and Indus River's RiverWorks.TM. VPN planned products. However, none of these products are known to offer capabilities which minimizes delay and session by a controlled fail-over process.

Brief Summary Text (6):
These VPNs place enormous demands on the enterprise network infrastructure. Single points of failure components such as gateways, firewalls, tunnel servers and other choke points that need to be made highly reliable and scaleable are being addressed with redundant equipment such as "hot standbys" and various types of clustering systems.

Brief Summary Text (7):
For example, CISCO.TM. Inc. now offers a new product called LocalDirector.TM. which functions as a front-end to a group of servers, dynamically load balances TCP traffic between servers to ensure timely access and response to requests. The LocalDirector provides the appearance, to end users, of a "virtual" server. For purposes of providing continuous access if the LocalDirector fails, users are required to purchase a redundant LocalDirector system which is directly attached to the primary unit, the redundant unit acting as a "hot" standby. The standby unit does no processing work itself until the master unit fails. The standby unit uses the failover IP address and the secondary Media Access Control (MAC) address (which are the same as the primary unit), thus no Address Resolution Protocol (ARP) is required to switch to the standby unit. However, because the standby unit does not keep state information on each connection, all active connections are dropped and must be re-established by the clients. Moreover, because the "hot standby" does no concurrent processing it offers no processing load relief nor scaling ability.

Drawing Description Text (7):
FIG. 5 illustrates a general memory map of the preferred embodiment of a cluster member acting as a tunnel-server.

Detailed Description Text (3):
cluster is disclosed. In the following description for purposes of explanation, specific data and configurations are set forth in order to provide a thorough understanding of the present invention. In the presently preferred embodiment the IP Network cluster is described in terms of a VPN tunnel-server cluster. However, it will be apparent to one skilled in these arts that the present invention may be practiced without the specific details, in various applications such as a firewall

cluster, a gateway or router cluster, etc. In other instances, well-known systems and protocols are shown and described in diagrammatical or block diagram form in order not to obscure the present invention unnecessarily.

Detailed Description Text (5):
The environment in which the present invention is used encompasses the general distributed computing scene which includes generally local area networks with hubs, routers, gateways, tunnel-servers, applications servers, etc. connected to other clients and other networks via the Internet, wherein programs and data are made available by various members of the system for execution and access by other members of the system. Some of the elements of a typical internet network configuration are shown in FIG. 1, wherein a number of client machines 105 possibly in a branch office of an enterprise, are shown connected to a Gateway/hub/tunnel-server/etc. 106 which is itself connected to the internet 107 via some internet service provider (ISP) connection 108. Also shown are other possible clients 101, 103 similarly connected to the internet 107 via an ISP connection 104, with these units communicating to possibly a home office via an ISP connection 109 to a gateway/tunnel-server 110 which is connected 111 to various enterprise application servers 112, 113, 114 which could be connected through another hub/router 115 to various local clients 116, 117, 118.

Detailed Description Text (14):
The present invention is an Internet Protocol (IP) clustering system which can provide a highly scalable system which optimizes-throughput by adaptively load balancing its components, and which minimizes delay and session loss by a controlled fail-over process. A typical IP cluster system of the preferred embodiment is shown in FIG. 4 wherein the internet 107 is shown connected to a typical IP cluster 401 which contains programmed general purpose computer units 403, 405, 407, 409 which act as protocol stack processors for message packets received. The IP cluster 401 is typically connected to application servers or other similar type units 411 in the network. In this figure it is shown that there may be any number of units in a cluster (e.g. member "n" 409) however for purposes of further illustration the cluster will be depicted as having three units, understanding that the cluster of the present invention is not limited to only three units. Also for purposes of illustration the preferred embodiment will be described as a cluster whose applications may be VPN tunnel protocols however it should be understood that this cluster invention may be used as a cluster whose application is to act as a Firewall, or to act as a gateway, or to act as a security device, etc.

Detailed Description Text (16):
"Point-to-Point Tunneling Protocol--PPTP", Glen Zorn, G. Pall, K. Hamzeh, W. Verthein, J. Taarud, W. Little, Jul. 28, 1998;

Detailed Description Text (17):
"Layer Two Tunneling Protocol", Allan Rubens, William Palter, T. Kolar, G. Pall, M. Littlewood, A. Valencia, K. Hamzeh, W. Verthein, J. Taarud, W. Mark Townsley, May 22, 1998;

Detailed Description Text (30):
Tunneling protocols such as the Point-to-Point Tunneling Protocol (PPTP) and Layer 2 Tunneling Protocol (L2TP) although currently only "drft" standards, are expected to be confirmed as official standards by the Internet Engineering Task Force (IETF) in the very near future, and these protocols together with the Internet Security Protocol (IPSec), provide the basis for the required security of these VPNs.

Detailed Description Text (50):
Basically the fields containing the IP addresses, IP protocol, and TCP/UDP port numbers, and if the application is L2TP, the session and tunnel ID fields are all added together (logical XOR) and then shifted to produce a unique "work unit"

number between 0 and 1023.

Detailed Description Text (65):
Having described the invention in terms of a preferred embodiment, it will be recognized by those skilled in the art that various types of general purpose computer hardware may be substituted for the configuration described above to achieve an equivalent result. Similarly, it will be appreciated that arithmetic logic circuits are configured to perform each required means in the claims for processing internet security protocols and tunneling protocols; for permitting the master unit to adaptively distribute processing assignments for incoming messages and for permitting cluster members to recognize which messages are theirs to process; and for recognizing messages from other members in the cluster. It will be apparent to those skilled in the art that modifications and variations of the preferred embodiment are possible, which fall within the true spirit and scope of the invention as measured by the following claims.

Current US Original Classification (1):
709/224

Encapsulation

First Hit Fwd Refs

Generate Collection

L14: Entry 6 of 12 File: USPT Aug 8, 2000

DOCUMENT-IDENTIFIER: US 6101543 A
TITLE: Pseudo network adapter for frame capture, encapsulation and encryption

Abstract Text (1):
A new pseudo network adapter is disclosed providing an interface for capturing packets from a local communications protocol stack for transmission on the virtual private network. The system further includes a Dynamic Host Configuration Protocol (DHCP) server emulator, and an Address Resolution Protocol (ARP) server emulator. The new system indicates to the local communications protocol stack that nodes on a remote private network are reachable through a gateway that is in turn reachable through the pseudo network adapter. The new pseudo network adapter includes a transmit path for processing data packets from the local communications protocol stack for transmission through the pseudo network adapter. The transmit path includes an encryption engine for encrypting the data packets and an encapsulation engine for encapsulating the encrypted data packets into tunnel data frames. The pseudo network adapter passes the tunnel data frames back to the local communications protocol stack for transmission to a physical network adapter on a remote server node. The new pseudo network adapter further includes an interface into a transport layer of the-local communications protocol stack for capturing received data packets from the remote server node, and a receive path for processing received data packets captured from the transport layer of the local communications protocol stack. The receive path includes a decapsulation engine, and a decryption engine, and passes the decrypted, decapsulated data packets back to the local communications protocol stack for delivery to a user.

Brief Summary Text (5):
One approach to providing secure communications is to form a virtual private network. In a virtual private network, secure communications are provided by encapsulating and encrypting messages. Encapsulated messaging in general is referred to as "tunneling". Tunnels using encryption may provide protected communications between users separated by a public network, or among a subset of users of a private network.

Brief Summary Text (9):
A tunnel may be implemented using a virtual or "pseudo" network adapter that appears to the communications protocol stack as a physical device and which provides a virtual private network. A pseudo network adapter must have the capability to receive packets from the communications protocol stack, and to pass received packets back through the protocol stack either to a user or to be transmitted.

Brief Summary Text (10):
A tunnel endpoint is the point at which any encryption/decryption and encapsulation/decapsulation provided by a tunnel is performed. In existing systems, the tunnel end points are pre-determined network layer addresses. The source network layer address in a received message is used to determine the "credentials" of an entity requesting establishment of a tunnel connection. For example, a tunnel server uses the source network layer address to determine whether a requested tunnel connection is authorized. The source network layer address is also used to determine which cryptographic key or keys to use to decrypt received messages.

Brief Summary Text (11):
Existing tunneling technology is typically performed by encapsulating encrypted network layer packets (also referred to as "frames") at the network layer. Such systems provide "network layer within network layer" encapsulation of encrypted messages. Tunnels in existing systems are typically between firewall nodes which have statically allocated IP addresses. In such existing systems, the statically allocated IP address of the firewall is the address of a tunnel end point within the firewall. Existing systems fail to provide a tunnel which can perform authorization based for an entity which must dynamically allocate its network layer address. This is especially problematic for a user wishing to establish a tunnel in a mobile computing environment, and who requests a dynamically allocated IP address from an Internet Service Provider (ISP).

Brief Summary Text (12):
Because existing virtual private networks are based on network layer within network layer encapsulation, they are generally only capable of providing connection-less datagram type services. Because datagram type services do not guarantee delivery of packets, existing tunnels can only easily employ encryption methods over the data contained within each transmitted packet. Encryption based on the contents of multiple packets is desirable, such as cipher block chaining or stream ciphering over multiple packets. For example, encrypted data would advantageously be formed based not only on the contents of the present packet data being encrypted, but also based on some attribute of the connection or session history between the communicating stations. Examples of encryption algorithms and keyed encryption are disclosed in many textbooks, for example "Applied Cryptography--Protocols, Algorithms, and Source Code in C", by Bruce Schneier, published by John Wiley and Sons, New York, N.Y. copyright 1994.

Brief Summary Text (13):
Thus there is required a new pseudo network adapter providing a virtual private network having a dynamically determined end point to support a user in a mobile computing environment. The new pseudo network adapter should appear to the communications protocol stack of the node as an interface to an actual physical device. The new pseudo network adapter should support guaranteed, in-order delivery of frames over a tunnel to conveniently support cipher block chaining mode or stream cipher encryption over multiple packets.

Brief Summary Text (17):
encapsulating the encrypted data packets into tunnel data frames. The pseudo network adapter passes the tunnel data frames back to the local communications protocol stack for transmission to a physical network adapter on a remote server node.

Brief Summary Text (18):
In a further aspect of the present system, the pseudo network adapter includes a digest value in a digest field in each of the tunnel data frames. A keyed hash function is a hash function which takes data and a shared cryptographic key as inputs and outputs a digital signature referred to as a digest. The value of the digest field is equal to an output of a keyed hash function applied to data consisting of the data packet encapsulated within the tunnel data frame concatenated with a counter value equal to a total number of tunnel data frames previously transmitted to the remote server node. In another aspect of the system, the pseudo network adapter processes an Ethernet header in each one of the captured data packets, including removing the Ethernet header.

Brief Summary Text (20):
Thus there is disclosed a new pseudo network adapter providing a virtual private network having dynamically determined end points to support users in a mobile computing environment. The new pseudo network adapter provides a system for

capturing a fully formed frame prior to transmission. The new pseudo network adapter appears to the communications protocol stack of the station as an interface to an actual physical device. The new pseudo network adapter further includes encryption capabilities to conveniently provide secure communications between tunnel end points using stream mode encryption or cipher block chaining over multiple packets.

Drawing Description Text (5):
FIG. 3 is a block diagram showing an example embodiment of a tunnel connection across a public network between two tunnel servers;

Drawing Description Text (6):
FIG. 4 is a flow chart showing an example embodiment of steps performed to establish a tunnel connection;

Drawing Description Text (7):
FIG. 5 is a flow chart showing an example embodiment of steps performed to perform session key management for a tunnel connection;

Drawing Description Text (13):
FIG. 11 is a state diagram showing an example embodiment of a state machine forming a tunnel connection in a network node initiating a tunnel connection;

Drawing Description Text (14):
FIG. 12 is a state diagram showing an example embodiment of a state machine forming a tunnel connection in a server computer;

Drawing Description Text (15):
FIG. 13 is a state diagram showing an example embodiment of a state machine forming a tunnel connection in a relay node;

Drawing Description Text (16):
FIG. 14 is a block diagram showing an example embodiment of a tunnel connection between a client computer (tunnel client) and a server computer (tunnel server);

Detailed Description Text (8):
FIG. 3 shows an example configuration of network nodes for which the presently disclosed system is applicable. In the example of FIG. 3, the tunnel server A is an initiator of the tunnel connection. As shown in FIG. 3, the term "tunnel relay" node is used to refer to a station which forwards data packets between transport layer connections (for example TCP connections).

Detailed Description Text (9):
For example, in the present system a tunnel relay may be dynamically configured to forward packets between transport layer connection 1 and transport layer connection 2. The tunnel relay replaces the header information of packets received over transport layer connection 1 with header information indicating transport layer connection 2. The tunnel relay can then forward the packet to a firewall, which may be conveniently programmed to pass packets received over transport layer connection 2 into a private network on the other side of the firewall. In the present system, the tunnel relay dynamically forms transport layer connections when a tunnel connection is established. Accordingly the tunnel relay is capable of performing dynamic load balancing or providing redundant service for fault tolerance over one or more tunnel servers at the time the tunnel connection is established.

Detailed Description Text (10):
FIG. 3 shows a Tunnel Server A 46 in a private network N1 48, physically connected with a first Firewall 50. The first Firewall 50 separates the private network N1 48 from a public network 52, for example the Internet. The first Firewall 50 is for example physically connected with a Tunnel Relay B 54, which in turn is virtually

connected through the public network 52 with a Tunnel Relay C. The connection between Tunnel Relay B and Tunnel Relay C may for example span multiple intervening forwarding nodes such as routers or gateways through the public network 52.

Detailed Description Text (11):
The Tunnel Relay C is physically connected with a second Firewall 58, which separates the public network 52 from a private network N2 60. The second Firewall 58 is physically connected with a Tunnel Server D 62 on the private network N2 60. During operation of the elements shown in FIG. 3, the Tunnel Server D 62 provides routing of IP packets between the tunnel connection with Tunnel Server A 46 and other stations on the private network N2 60. In this way the Tunnel Server D 62 acts as a router between the tunnel connection and the private network N2 60.

Detailed Description Text (12):
During operation of the elements shown in FIG. 3, the present system establishes a tunnel connection between the private network N1 48 and the private network N2 60. The embodiment of FIG. 3 thus eliminates the need for a dedicated physical cable or line to provide secure communications between the private network 48 and the private network 60. The tunnel connection between Tunnel Server A 46 and Tunnel Server D 62 is composed of reliable, pair-wise transport layer connections between Tunnel Server A 46 (node "A"), Tunnel Relay B 54 (node "B"), Tunnel Relay C 56 (node "C"), and Tunnel Server D 62 (node "D"). For example, such pair-wise connections may be individual transport layer connections between each node A and node B, node B and node C, and node C and node D. In an alternative embodiment, as will be described below, a tunnel connection may alternatively be formed between a stand-alone PC in a public network and a tunnel server within a private network.

Detailed Description Text (13):
FIG. 4 and FIG. 5 show an example embodiment of steps performed during establishment of the tunnel connection between Tunnel Server A 46 (node "A") and Tunnel Server D 62 (node "D") as shown in FIG. 3. Prior to the steps shown in FIG. 4, node A selects a tunnel path to reach node D. The tunnel path includes the tunnel end points and any intervening tunnel relays. The tunnel path is for example predetermined by a system administrator for node A. Each tunnel relay along the tunnel path is capable of finding a next node in the tunnel path, for example based on a provided next node name (or "next node arc"), using a predetermined naming convention and service, for example the Domain Name System (DNS) of the TCP/IP protocol suite.

Detailed Description Text (15):
resolve the node name of the next node in the tunnel path, for example as found in a tunnel relay frame;

Detailed Description Text (16):
establish a reliable transport layer (TCP) connection to the next node in the tunnel path;

Detailed Description Text (17):
forward the tunnel relay frame down the newly formed reliable transport layer connection to the next node in the tunnel path.

Detailed Description Text (18):
As shown for example in FIG. 4, at step 70 node A establishes a reliable transport layer connection with node B. At step 72 node A identifies the next downstream node to node B by sending node B a tunnel relay frame over the reliable transport layer connection between node A and node B. The tunnel relay frame contains a string buffer describing all the nodes along the tunnel path (see below description of an example tunnel relay frame format). At step 74, responsive to the tunnel relay frame from node A, node B searches the string buffer in the relay frame to determine if the string buffer includes node B's node name. If node B finds its

node name in the string buffer, it looks at the next node name in the string buffer to find the node name of the next node in the tunnel path.

Detailed Description Text (19):
Node B establishes a reliable transport layer connection with the next node in the tunnel path, for example node C. Node B further forms an association between the reliable transport layer connection between Node A and Node B, over which the relay frame was received, and the newly formed reliable transport layer connection between Node B and Node C, and as a result forwards subsequent packets received over the reliable transport layer connection with Node A onto the reliable transport layer connection with Node C, and vice versa. At step 76 node B forwards the tunnel relay frame on the newly formed reliable transport layer connection to node C.

Detailed Description Text (20):
At step 78, responsive to the relay frame forwarded from node B, node C determines that the next node in the tunnel path is the last node in the tunnel path, and accordingly is a tunnel server. Node C may actively determine whether alternative tunnel servers are available to form the

Detailed Description Text (21):
tunnel connection. Node C may select one of the alternative available tunnel servers to form the tunnel connection in order to provide load balancing or fault tolerance. As a result node C may form a transport layer connections with one of several available tunnel servers, for example a tunnel server that is relatively underutilized at the time the tunnel connection is established. In the example embodiment, node C establishes a reliable transport layer connection with the next node along the tunnel path, in this case node D.

Detailed Description Text (23):
FIG. 5 shows an example of tunnel end point authentication and sharing of key exchange material provided by the present system. The present system supports passing authentication data and key exchange material through the reliable transport layer connections previously established on the tunnel path. The following are provided by use of a key exchange/authentication REQUEST frame and a key exchange/authentication RESPONSE frame:

Detailed Description Text (24):
a) mutual authentication of both endpoints of the tunnel connection;

Detailed Description Text (25):
b) establishment of shared session encryption keys and key lifetimes for encrypting/authenticating subsequent data sent through the tunnel connection;

Detailed Description Text (27):
e) exchange of any other connection-specific data between the tunnel endpoints, for example strength and type of cipher to be used, any compression of the data to be used, etc. This data can also be used by clients of this protocol to qualify the nature of the authenticated connection.

Detailed Description Text (28):
At step 90 a key exchange/authentication request frame is forwarded over the reliable transport layer connections formed along the tunnel path from node A to node D. At step 92, a key exchange/authentication response frame is forwarded from node D back to node A through the reliable transport layer connections. The attributes exchanged using the steps shown in FIG. 5 may be used for the lifetime of the tunnel connection. In an alternative embodiment the steps shown in FIG. 5 are repeated as needed for the tunnel end points to exchange sufficient key exchange material to agree upon a set of session parameters for use during the tunnel connection such as cryptographic keys, key durations, and choice of

encryption/decryption algorithms.

Detailed Description Text (29):
Further in the disclosed system, the names used for authentication and access control with regard to node A and node D need not be the network layer address or physical address of the nodes. For example, in an alternative embodiment where the initiating node sending the tunnel relay frame is a stand-alone PC located within a public network, the user's name may be used for authentication and/or access control purposes. This provides a significant improvement over existing systems which base authorization on predetermined IP addresses.

Detailed Description Text (30):
FIG. 6 shows the format of an example embodiment of a tunnel relay frame. The tunnel frame formats shown in FIGS. 6, 7, 8 and 9 are encapsulated within the data portion of a transport layer (TCP) frame when transmitted. Alternatively, another equivalent, connection-oriented transport layer protocol having guaranteed, in-sequence frame delivery may be used. The example TCP frame format, including TCP header fields, is conventional and not shown.

Detailed Description Text (31):
The field 100 contains a length of the frame. The field 102 contains a type of the frame, for example a type of RELAY. The field 104 contains a tunnel protocol version number. The field 106 contains an index into a string buffer field 112 at which a name of the originating node is located, for example a DNS host name of the node initially issuing the relay frame (node A in FIG. 3). The fields following the origin index field 106 contain indexes into the string buffer 112 at which names of nodes along the tunnel path are located. For example each index may be the offset of a DNS host name within the string buffer 112. In this way the field 108 contains the index of the name of the first node in the tunnel path, for example node B (FIG. 3). The field 110 contains the index of the name of the second node in the tunnel path, etc. The field 112 contains a string of node names of nodes in the tunnel path.

Detailed Description Text (32):
During operation of the present system, the initiating node, for example node A as shown in FIG. 3, transmits a tunnel relay frame such as the tunnel relay frame shown in FIG. 6. Node A sends the tunnel relay frame to the first station along the tunnel path, for example node B (FIG. 3), over a previously established reliable transport layer connection. Node B searches the string buffer in the tunnel relay frame to find its node name, for example its DNS host name. Node B finds its node name in the string buffer indexed by path index 0, and then uses the contents of path index 1 110 to determine the location within the string buffer 112 of the node name of the next node along the tunnel path. Node B uses this node name to establish a reliable transport layer connection with the next node along the tunnel path. Node B then forwards the relay frame to the next node. This process continues until the end node of the tunnel route, for example tunnel server D 62 (FIG. 3) is reached.

Detailed Description Text (33):
FIG. 7 shows the format of an example embodiment of a key exchange/key authentication request frame. The field 120 contains a length of the frame. The field 122 contains a type of the frame, for example a type of REQUEST indicating a key exchange/key authentication request frame. The field 124 contains a tunnel protocol version number. The field 126 contains an offset of the name of the entity initiating the tunnel connection, for example the name of a user on the node originally issuing the request frame. This name and key exchange material in the request frame are used by the receiving tunnel end point to authenticate the key exchange/authentication REQUEST. The name of the entity initiating the tunnel connection is also use to authorize any subsequent tunnel connection, based on predetermined security policies of the system. The field 128 contains an offset

into the frame of the node name of the destination node, for example the end node of the tunnel shown as node D 62 in FIG. 3.

Detailed Description Text (34):
The field 130 contains an offset into the frame at which key exchange data as is stored, for example within the string buffer field 138. The key exchange data for example includes key exchange material used to determine a shared set of encryption parameters for the life of the tunnel connection such as cryptographic keys and any validity times associated with those keys. The key exchange data, as well as the field 132, further include information regarding any shared set of cryptographic transforms to be used and any other connection-specific parameters, such as strength and type of cipher to be used, type of compression of the data to be used, etc. The field 134 contains flags, for example indicating further information about the frame. The field 136 contains client data used in the tunnel end points to configure the local routing tables so that packets for nodes reachable through the virtual private network are sent through the pseudo network adapters. In an example embodiment, the string buffer 138 is encrypted using a public encryption key of the receiving tunnel end point.

Detailed Description Text (35):
During operation of the present system, one of the end nodes of the tunnel sends a key exchange/authentication REQUEST frame as shown in FIG. 7 to the other end node of the tunnel in order to perform key exchange and authentication as described in step 90 of FIG. 5.

Detailed Description Text (36):
FIG. 8 shows the format of an example embodiment of a key exchange/key authentication response frame, referred to as a connection RESPONSE frame. The field 150 contains a length of the frame. The field 152 contains a type of the frame, for example a type of connection RESPONSE indicating a key exchange/key authentication request frame. The field 154 contains a tunnel protocol version number.

Detailed Description Text (37):
The field 156 contains an offset into the frame at which key exchange data as is stored, for example within the string buffer field 163. The key exchange data for example includes key exchange material to be used for encryption/decryption over the life of the tunnel connection and any validity times associated with that key exchange material. The key exchange data, as well as the field 158, further includes information regarding any shared set of cryptographic transforms to be applied to subsequent data and any other connection-specific parameters, such as strength and type of cipher to be used, any compression of the data to be used, etc. The field 160 contains flags, for example indicating other information about the frame. The client data field 162 contains data used by the pseudo network adapters in the tunnel end points to configure the local routing tables so that packets for nodes in the virtual private network are sent through the pseudo network adapters. The string buffer includes key exchange material. The string buffer is for example encrypted using a public encryption key of the receiving tunnel end point, in the this case the initiator of the tunnel connection.

Detailed Description Text (38):
During operation of the present system, one of the end nodes of the tunnel sends a key exchange/authentication RESPONSE frame as shown in FIG. 7 to the other end node of the tunnel in order to perform key exchange and authentication as described in step 92 of FIG. 5.

Detailed Description Text (39):
FIG. 9 shows the format of an example embodiment of an tunnel data frame used to communicate through a tunnel connection. FIG. 9 shows how an IP datagram may be encapsulated within a tunnel frame by the present system for secure communications

through a virtual private network. The field 170 contains a length of the frame. The field 172 contains a type of the frame, for example a type of DATA indicating a tunnel data frame. The field 174 contains a tunnel protocol version number.

Detailed Description Text (41):
The data frame format includes a digital signature generated by the transmitting tunnel end point referred to as a "digest". The value of the digest ensures data integrity, for example by detecting invalid frames and replays of p reviously transmitted valid frames. The digest is the output of a conventional keyed cryptographic hash function applied to both the encapsulated datagram 190 and a monotonically increasing sequence number. The resulting hash output is passed as the value of the digest field 187. The sequence number is not included in the data frame. In the example embodiment, the sequence number is a counter maintained by the transmitter (for example node A in FIG. 3) of all data frames sent to the receiving node (for example node D in FIG. 3) since establishment of the tunnel connection.

Detailed Description Text (42):
In order to determine if the data frame is invalid or a duplicate, the receiving node decrypts the encapsulated datagram 190, and applies the keyed cryptographic hash function (agreed to by the tunnel end nodes during the steps shown in FIG. 5) to both the decrypted encapsulated datagram and the value of a counter indicating the number of data frames received from the transmitter since establishment of the tunnel connection. For example the keyed hash function is applied to the datagram concatenated to the counter value. If the resulting hash output matches the value of the digest field 187, then the encapsulated datagram 190 was received correctly and is not a duplicate. If the hash output does not match the value of the digest field 187, then the integrity check fails, and the tunnel connection is closed. The field 188 contains an encrypted network layer datagram, for example an encrypted IP datagram.

Detailed Description Text (43):
The encapsulated datagram may be encrypted using various encryption techniques. An example embodiment of the present system advantageously encrypts the datagram 190 using either a stream cipher or cipher block chaining encryption over all data transmitted during the life of the tunnel connection. This is enabled by the reliable nature of the transport layer connections within the tunnel connection. The specific type of encryption and any connection specific symmetric encryption keys used is determined using the steps shown in FIG. 5. The fields in the tunnel data frame other than the encapsulated datagram 188 are referred to as the tunnel data frame header fields.

Detailed Description Text (44):
FIG. 10 is a block diagram showing an example embodiment of a "close connection" frame. The field 190 contains the length of the frame. The field 191 contains a frame type, for example having a value equal to CLOSE. Field 192 contains a value equal to the current protocol version number of the tunnel protocol. The field 193 contains a status code indicating the reason the tunnel connection is being closed.

Detailed Description Text (45):
During operation of the present system, when end point of a tunnel connection determines that the tunnel connection should be closed, a close connection frame as shown in FIG. 10 is transmitted to the other end point of the tunnel connection. When a close connection close frame is received, the receiver closes the tunnel connection and no further data will be transmitted or received through the tunnel connection.

Detailed Description Text (46):
FIG. 11 is a state diagram showing an example embodiment of forming a tunnel

connection in a node initiating a tunnel connection. In FIG. 11, FIG. 12, and FIG. 13, states are indicated by ovals and actions or events are indicated by rectangles. For example the tunnel server node A as shown in FIG. 3 may act as a tunnel connection initiator when establishing a tunnel connection with the tunnel server node D. Similarly the client system 247 in FIG. 14 may act as a tunnel connection initiator when establishing a tunnel connection with the tunnel server. The tunnel initiator begins in an idle state 194. Responsive to an input from a user indicating that a tunnel connection should be established, the tunnel initiator transitions from the idle state 194 to a TCP Open state 195. In the TCP Open state 195, the tunnel initiator establishes a reliable transport layer connection with a first node along the tunnel path. For example, the tunnel initiator opens a socket interface associated with a TCP connection to the first node along the tunnel path. In FIG. 3 node A opens a socket interface associated with a TCP connection with node B.

Detailed Description Text (47):
Following establishment of the reliable transport layer connection in the TCP Open state 195, the tunnel initiator enters a Send Relay state 197. In the Send Relay state 197, the tunnel initiator transmits a relay frame at 198 over the reliable transport layer connection. Following transmission of the relay frame, the tunnel initiator enters the connect state 199. If during transmission of the relay frame there is a transmission error, the tunnel initiator enters the Network Error state 215 followed by the Dying state 208. In the Dying state 208, the tunnel initiator disconnects the reliable transport layer connection formed in the TCP Open state 195, for example by disconnecting a TCP connection with Node B. Following the disconnection at 209, the tunnel initiator enters the Dead state 210. The tunnel initiator subsequently transitions back to the Idle state 194 at a point in time predetermined by system security configuration parameters.

Detailed Description Text (48):
In the Connect state 199, the tunnel initiator sends a key exchange/authentication REQUEST frame at 200 to the tunnel server. Following transmission of the key exchange/authentication REQUEST frame 200, the tunnel initiator enters the Response Wait state 201. The tunnel initiator remains in the Response Wait state 201 until it receives a key exchange/authentication RESPONSE frame 202 from the tunnel server. After the key exchange/authentication RESPONSE frame is received at 202, the tunnel initiator enters the Authorized state 203, in which it may send or receive tunnel data frames. Upon receipt of a CLOSE connection frame at 216 in the Authorized state 203, the tunnel initiator transitions to the Dying state 208.

Detailed Description Text (49):
Upon expiration of a session encryption key at 211, the tunnel initiator

Detailed Description Text (50):
enters the Reconnect state 212, and sends a CLOSE connection frame at 213 and disconnects the TCP connection with the first node along the tunnel path at 214. Subsequently the tunnel initiator enters the TCP Open state 195.

Detailed Description Text (51):
If during the authorized state 203, a local user issues an End Session command at 204, or there is a detection of an authentication or cryptography error in a received data frame at 205, the tunnel initiator enters the Close state 206. During the Close state 206 the tunnel initiator sends a CLOSE connection frame at 207 to the tunnel server. The tunnel initiator then enters the Dying state at 208.

Detailed Description Text (52):
FIG. 12 is a state diagram showing the states within an example embodiment of a tunnel server, for example node D in FIG. 3 or tunnel server 253 in FIG. 14. The tunnel server begins in an Accept Wait state 217. In the Accept Wait state 217, the tunnel server receives a request for a reliable transport layer connection, for

example a TCP connection request 218 from the last node in the tunnel path prior to the tunnel server, for example Node C in FIG. 3. In response to a TCP connection request 218 the tunnel server accepts the request and establishes a socket interface associated with the resulting TCP connection with Node C.

Detailed Description Text (53):
Upon establishment of the TCP connection with the last node in the tunnel path prior to the tunnel server, the tunnel server enters the Receive Relay state 219. In the Receive Relay state 219, the tunnel server waits to receive a relay frame at 220, at which time the tunnel server enters the Connect Wait state 221. If there is some sort of network error 234 during receipt of the relay frame at 219, the tunnel server enters the Dying state 230. During the Dying state 230 the tunnel server disconnects at 231 the transport layer connection with the last node in the tunnel path prior to the tunnel server. After disconnecting the connection, the tunnel server enters the Dead state 232.

Detailed Description Text (54):
In the Connect Wait state 221, the tunnel server waits for receipt of a key exchange/authentication REQUEST frame at 222. Following receipt of the key exchange/authentication REQUEST frame at 222, the tunnel server determines whether the requested tunnel connection is authorized at step 223. The determination of whether the tunnel connection is authorized is based on a name of the tunnel initiator, and the key exchange material within the key exchange/authentication REQUEST frame.

Detailed Description Text (55):
If the requested tunnel connection is authorized the tunnel server sends a key exchange/authentication RESPONSE frame at 224 back to the tunnel initiator. If the requested tunnel connection is not authorized, the tunnel server enters the Close state 228, in which it sends a close connection frame at 229 to the tunnel client. Following transmission of the CLOSE connection frame at 229, the tunnel server enters the Dying state 230.

Detailed Description Text (56):
If the requested tunnel connection is determined to be authorized at step 223, the tunnel server enters the Authorized state 225. In the Authorized state, the tunnel server transmits and receives tunnel data frames between itself and the tunnel initiator. If during the Authorized state 225, the tunnel server receives a CLOSE connection frame at 233, the tunnel server transitions to the Dying state 230. If during the authorized state 225, the tunnel server receives an end session command from a user at 226, then the tunnel server transitions to the Close state 228, and transmits a close connection frame at 229 to the tunnel initiator. If the tunnel server in the Authorized state 225 detects an integrity failure in a received packet, the tunnel server transitions to the Close state 228. In the close state 228 the tunnel server sends a CLOSE connection frame at 229 and subsequently enters the Dying state 230.

Detailed Description Text (57):
FIG. 13 is a state diagram showing an example embodiment of a state machine within a tunnel relay node. The tunnel relay node begins in an Accept Wait state 235. When a request is received to form a reliable transport layer connection at 236, a reliable transport layer connection is accepted with the requesting node. For example, a TCP connection is accepted between the relay node and the preceding node in the tunnel path.

Detailed Description Text (58):
The relay node then transitions to the Receive Relay state 237. During the Receive Relay state 237, the relay node receives a relay frame at 238. Following receipt of the relay frame at 238, the relay node determines what forwarding address should be used to forward frames received from the TCP connection established responsive to

the TCP connect event 236. If the next node in the tunnel path is a tunnel server, the forwarding address may be selected at 239 so as to choose an underutilized tunnel server from a group of available tunnel servers or to choose an operational server where others are not operational.

Detailed Description Text (60):
Following establishment of the new connection at event 241, the tunnel relay enters the Forward state 242. During the Forward state 242, the relay node forwards all frames between the connection established at 236 and those connections established at 241. Upon detection of a network error or receipt of a frame indicating a closure of the tunnel connection at 243, the tunnel relay enters the Dying state 244. Following the Dying state 244, the relay node disconnects any connections established at event 241. The relay node then enters the Dead state 246.

Detailed Description Text (61):
FIG. 14 shows an example embodiment of a virtual private network 249 formed by a pseudo network adapter 248 and a tunnel connection between a tunnel client 247 and a tunnel server 253 across a public network 251. The tunnel server 253 and tunnel client 247 are for example network stations including a CPU or microprocessor, memory, and various I/O devices. The tunnel server 253 is shown physically connected to a private LAN 256 including a Network Node 1 257 and a Network Node 2 258, through a physical network adapter 254. The tunnel server 253 is further shown physically connected with a firewall 252 which separates the private LAN 256 from the public network 251. The firewall 252 is physically connected with the public network 251. The tunnel server 253 is further shown including a pseudo network adapter 255. The client system 247 is shown including a physical network adapter 250 physically connected to the public network 251.

Detailed Description Text (62):
During operation of the elements shown in FIG. 14, nodes within the virtual private network 249 appear to the tunnel client 247 as if they were physically connected to the client system through the pseudo network adapter 248. Data transmissions between the tunnel client and any nodes that appear to be within the virtual private network are passed through the pseudo network adapter 248. Data transmissions between the tunnel client 247 and the tunnel server 253 are physically accomplished using a tunnel connection between the tunnel client 247 and the tunnel server 253.

Detailed Description Text (63):
FIG. 15 shows elements in an example embodiment of a pseudo network adapter such as the pseudo network adapter 248 in FIG. 14. In an example embodiment the elements shown in FIG. 15 are implemented as software executing on the tunnel client 247 as shown in FIG. 14. In FIG. 15 there is shown a pseudo network adapter 259 including a virtual adapter driver interface 263, an encapsulation engine 264, an encryption engine 265, a decapsulation engine 268, and a decryption engine 266. Further shown in the pseudo network adapter 259 are an ARP server emulator 270 and a Dynamic Host Configuration Protocol (DHCP) server emulator.

Detailed Description Text (65):
During operation of the elements shown in FIG. 15, the pseudo network adapter 259 registers with the network layer in the TCP/IP stack 260 that it is able to reach the IP addresses of nodes within the virtual private network 249 as shown in FIG. 14. For example, the pseudo network adapter on the client system registers that it can reach the pseudo network adapter on the server. Subsequently, a message from the tunnel client addressed to a node reachable through the virtual private network will be passed by the TCP/IP stack to the pseudo network adapter 259. The pseudo network adapter 259 then encrypts the message, and encapsulates the message into a tunnel data frame. The pseudo network adapter 259 then passes the tunnel data frame back to the TCP/IP protocol stack 260 to be sent through to the physical network adapter in the tunnel server. The tunnel server passes the received data frame to

the pseudo network adapter in the server, which de-encapsulates and decrypts the message.

Detailed Description Text (66):
FIG. 16 shows a more detailed example embodiment of a pseudo network adapter 280. The pseudo network adapter 280 includes a virtual network adapter driver interface 288. The transmit path 290 includes an encryption engine 292, and an encapsulation engine 294. The encapsulation engine 294 is interfaced with a TCP/IP transmit interface 312 within a TCP/IP protocol stack, for example a socket interface associated with the first relay node in the tunnel path, or with the remote tunnel end point if the tunnel path includes no relays.

Detailed Description Text (68):
Further shown in the pseudo network adapter 280 of FIG. 16 is a receive path 296 having a decryption engine 298 interfaced to the virtual network adapter interface 288 and a decapsulation engine 300. The decapsulation engine 300 in turn is interfaced to a TCP/IP receive function 314 in the TCP/IP protocol stack 282, for example a socket interface associated with the first relay in the tunnel path, or with the remote tunnel end point if the tunnel path includes no relays. The pseudo network adapter 280 further includes an ARP server emulator 304 and a DHCP server emulator 306. ARP and DHCP request packets 302 are passed to the ARP server emulator 304 and DHCP server emulator 306 respectively. When a received packet is passed from the receive path 296 to the TCP/IP stack 282, a receive event must be indicated to the TCP/IP stack 282, for example through an interface such the Network Device Interface Specification (NDIS), defined by Microsoft.TM. Corporation.

Detailed Description Text (70):
During operation of the elements shown in FIG. 16, a user passes data for transmission to the TCP/IP protocol stack 282, and indicates the IP address of the node to which the message is to be transmitted, for example through a sock et interface to the TCP layer. The TCP/IP protocol stack 282 then determines whether the destination node is reachable through the virtual private network. If the message is for a node that is reachable through the virtual private network, the TCP/IP protocol stack 282 an ethernet packet 286 corresponding to the message to the pseudo network adapter 280. The pseudo network adapter 280 then passes the ethernet packet 286 through the transmit path, in which the ethernet packet is encrypted and encapsulated into a tunnel data frame. The tunnel data frame is passed back into the TCP/IP protocol stack 282 through the TCP/IP transmit function 312 to be transmitted to the tunnel server through the tunnel connection. In an example embodiment, a digest value is calculated for the tunnel data frame before encryption within the transmit path within the pseudo network adapter.

Detailed Description Text (71):
Further during operation of the elements shown in FIG. 16, when the TCP/IP protocol stack 282 receives a packet from the remote endpoint of the TCP/IP tunnel connection, for example the tunnel server, the packet is passed to the pseudo network adapter 280 responsive to a TCP receive event. The pseudo network adapter 280 then decapsulates the packet by removing the tunnel header. The pseudo network adapter further decrypts the decapsulated data and passes it back to the TCP/IP protocol stack 282. The data passed from the pseudo network adapter 280 appears to the TCP/IP protocol stack 282 as an ethernet packet received from an actual physical device, and is the data it contains is passed on to the appropriate user by the TCP/IP protocol stack 282 based on information in the ethernet packet header provided by the pseudo network adapter.

Detailed Description Text (72):
FIG. 17 is a flow chart showing steps performed by an example embodiment of a pseudo network adapter during packet transmission, such as in the transmit path 290 of FIG. 14. The TCP/IP protocol stack determines that the destination node of a

packet to be transmitted is reachable through the virtual LAN based on the destination IP address of the packet and a network layer routing table. At step 320 the packet is passed to the pseudo network adapter from the TCP/IP protocol stack. As a result, a send routine in the pseudo adapter is triggered for example in the virtual network adapter interface 288 of FIG. 16.

Detailed Description Text (73):
At step 322 the pseudo network adapter send routine processes the Ethernet header of the packet provided by the TCP/IP stack, and removes it. At step 324, the send routine determines whether the packet is an ARP request packet. If the packet is an ARP request packet for an IP address of a node on the virtual LAN, such as the pseudo network adapter of the tunnel server, then step 324 is followed by step 326. Otherwise, step 324 is followed by step 330.

Detailed Description Text (74):
At step 326, the ARP server emulator in the pseudo network adapter generates an ARP reply packet. For example, if the ARP request were for a physical address corresponding to the IP address of the pseudo network adapter on the tunnel server, the ARP reply would indicate a predetermined, reserved physical address to be associated with that IP address. At step 328 the pseudo network adapter passes the ARP response to the virtual network adapter interface. The virtual network adapter interface then indicates a received packet to the TCP/IP protocol stack, for example using an NDIS interface. The TCP/IP protocol stack then processes the ARP response as if it had been received over an actual physical network.

Detailed Description Text (76):
At step 334, the DHCP server emulator in the pseudo network adapter generates a DHCP response. The format of DHCP is generally described in the DHCP RFC. At step 328 the pseudo network adapter passes the DHCP response to the virtual network adapter interface, for example indicating an IP address received from the tunnel server in the client data field of the key exchange/authentication RESPONSE frame. The virtual network adapter interface then indicates a received packet to the TCP/IP protocol stack. The TCP/IP protocol stack then processes the DHCP response as if it had been received over an actual physical network.

Detailed Description Text (77):
At step 334 the pseudo network adapter encrypts the message using an encryption engine such that only the receiver is capable of decrypting and reading the message. At step 336 the pseudo network adapter encapsulates the encrypted message into a tunnel data frame. At step 338 the pseudo network adapter transmits the tunnel data frame through the tunnel connection using the TCP/IP protocol stack.

Detailed Description Text (78):
FIG. 18 is a flow chart showing steps performed by an example embodiment of a pseudo network adapter during packet receipt, such as in the receive path 296 of FIG. 14. At step 350, the pseudo network adapter is notified that a packet has been received over the tunnel connection. At step 352

Detailed Description Text (79):
the pseudo network adapter decapsulates the received message by removing the header fields of the tunnel data frame. At step 354 the pseudo network adapter decrypts the decapsulated datagram from the tunnel data frame. At step 356, in an example embodiment, the pseudo network adapter forms an Ethernet packet from the decapsulated message. At step 358 the pseudo network adapter indicates that an Ethernet packet has been received to the TCP/IP protocol stack through the virtual network adapter interface. This causes the TCP/IP protocol stack to behave as if it had received an Ethernet packet from an actual Ethernet adapter.

Detailed Description Text (82):
At step 4 384, the pseudo network adapter 382 encrypts the data packets and

encapsulates them into tunnel data frames. The pseudo network adapter 382 then passes the tunnel data frames packets back to the TCP protocol layer 372 within the TCP/IP protocol stack through a conventional socket interface to the tunnel connection with the first node in the tunnel path.

Detailed Description Text (83):
The TCP protocol layer 372 then forms a TCP layer packet for each tunnel data frame, having the tunnel data frame as its data. The TCP frames are passed to the IP layer 374. At step 5 386 the routing table 378 is again searched, and this time the destination IP address is the IP address associated with the physical network adapter on the tunnel server, and accordingly is determined to be reachable over the physical network adapter 390. Accordingly at step 6 388 the device driver 390 for the physical network adapter is called to pass the packets to the physical network adapter. At step 7 392 the physical network adapter transmits the data onto the physical network 394.

Detailed Description Text (87):
In the example embodiment of FIG. 19, the pseudo network adapter 459 passes the data to a tunnel application program 466. The tunnel application program 466 encrypts the IP packet received from the IP layer and encapsulates it into a tunnel data frame. The tunnel application then passes the tunnel data frame including the encrypted data to the WinSock interface 452, indicating a destination IP address of the remote tunnel end point. The tunnel data frame is then passed through the TCP layer 454, IP layer 456, NDIS MAC layer interface 458, and physical layer 468, and transmitted on the network 470. Since the resulting packets do not contain a destination IP address which the pseudo network adapter has registered to convey, these packets will not be diverted to the pseudo network adapter.

Detailed Description Text (90):
In the example embodiment of FIG. 22, the pseudo network adapter 480 passes IP datagrams to be transmitted to a UNIX Daemon 486 associated with the tunnel connection. The UNIX Daemon 486 encrypts the IP packet(s) received from the IP layer 478 and encapsulates them into tunnel data frames. The UNIX daemon 486 then passes the tunnel data frames to the UNIX socket layer 474, through a socket associated with the tunnel connection. The tunnel data frames are then processed by the TCP layer 476, IP layer 478, data link layer 488, and physical layer 490 to be transmitted on the network 492. Since the resulting packets are not addressed to an IP address which the pseudo network adapter 480 has registered to convey, the packets will not be diverted to the pseudo network adapter 480.

Detailed Description Text (91):
FIG. 23 is a flow chart showing steps to initialize a example embodiment of a virtual private network. The steps shown in FIG. 23 are performed for example in the tunnel client 247 as shown in FIG. 14. At step 500 a tunnel application program executing in the tunnel client sends a tunnel relay frame to the tunnel server. At step 502 the tunnel application program sends a tunnel key exchange/authentication REQUEST frame to the tunnel server. The tunnel application in the tunnel server ignores the contents of the client data field in the tunnel key exchange/authentication REQUEST frame. The tunnel application in the tunnel server fills in the client data field in the tunnel key exchange/authentication RESPONSE frame with Dynamic Host Configuration Protocol (DHCP) information, for example including the following information in standard DHCP format:

Detailed Description Text (92):
1) IP Address for tunnel client Pseudo Network Adapter

Detailed Description Text (93):
2) IP Address for tunnel server Pseudo Network Adapter

Detailed Description Text (94):

3) Routes to nodes on the private network physically connected to the tunnel server which are to be reachable over the tunnel connection.

Detailed Description Text (95):
At step 504 the tunnel application receives a tunnel key exchange/authentication RESPONSE frame from the tunnel server. The client data field 508 in the tunnel connection response is made available to the pseudo network adapter in the tunnel client. The tunnel application in the tunnel client tells the TCP/IP stack that the pseudo network adapter in the tunnel client is active. The pseudo network adapter in the tunnel client is active and ready to be initialized at step 510.

Detailed Description Text (96):
The tunnel client system is configured such that it must obtain an IP address for the tunnel client pseudo network adapter dynamically. Therefore the TCP/IP stack in the tunnel client broadcasts a DHCP request packet through the pseudo network adapter. Accordingly, at step 512 the pseudo network adapter in the client receives a conventional DHCP request packet from the TCP/IP stack requesting a dynamically allocated IP address to associate with the pseudo network adapter. The pseudo network adapter passes the DHCP request packet to the DHCP server emulator within the pseudo network adapter, which forms a DHCP response based on the client data 508 received from the tunnel application. The DHCP response includes the IP address for the client pseudo adapter provided by the tunnel server in the client data. At step 514 the pseudo network adapter passes the DHCP response to the TCP/IP stack.

Detailed Description Text (97):
At step 520, the tunnel application modifies the routing tables within the tunnel client TCP/IP stack to indicate that the routes to the nodes attached to the private network to which the tunnel server is attached all are reachable only through the pseudo network adapter in the tunnel server. The IP address of the pseudo network adapter in the tunnel server provided in the client data is in this way specified as a gateway to the nodes on the private network to which the tunnel server is attached. In this way those remote nodes are viewed by the TCP/IP stack as being reachable via the virtual private network through the client pseudo network adapter.

Detailed Description Text (98):
At step 516 the pseudo network adapter in the tunnel client receives an ARP request for a physical address associated with the IP address of the pseudo network adapter in the tunnel server. The pseudo network adapter passes the ARP request to the ARP server emulator, which forms an ARP reply indicating a reserved physical address to be associated with the IP address of the pseudo network adapter in the tunnel server. At step 518 the pseudo network adapter passes the ARP response to the TCP/IP stack in the tunnel client. In response to the ARP response, the TCP/IP stack determines that packets addressed to any node on the virtual private network must be initially transmitted through the pseudo network adapter.

Current US Original Classification (1):
709/229

Current US Cross Reference Classification (1):
709/225

Current US Cross Reference Classification (2):
709/226

Current US Cross Reference Classification (3):
709/227

Current US Cross Reference Classification (4):
709/228

Current US Cross Reference Classification (5):
709/236

Current US Cross Reference Classification (6):
709/238

Current US Cross Reference Classification (7):
709/239

CLAIMS:

6. The pseudo network adapter of claim 1, further comprising:

a transmit path for processing data packets captured by said interface for capturing packets from said local communications protocol stack for transmission on said virtual private network;

an encryption engine, within said transmit path, for encrypting said data packets;

an encapsulation engine, within said transmit path, for encapsulating said encrypted data packets into tunnel data frames; and

a means for passing said tunnel data frames back to said local communications protocol stack for transmission to a physical network adapter on said remote server node.

7. The pseudo network adapter of claim 6, wherein said transmit path further includes means for storing a digest value in a digest field in

each of said tunnel data frames, said digest value equal to an output of a keyed hash function applied to said data packet encapsulated within said tunnel data frame concatenated with a counter value equal to a total number of tunnel data frames previously transmitted to said remote server node.

10. The pseudo network adapter of claim 9, further comprising:

a receive path for processing received data packets captured by said interface into said transport layer of said local communications protocol stack for capturing received data packets from said remote server node;

an decapsulation engine, within said receive path, for decapsulating said received data packets by removing a tunnel frame header; and

an decryption engine, within said receive path, for decrypting said received data packets;

a means for passing said received data packets back to said local communications protocol stack for delivery to a user.

17. The method of claim 12, further comprising:

processing data packets captured by said interface for capturing packets from said local communications protocol stack for transmission on said virtual private network in a transmit data path;

encrypting said data packets in an encryption engine, within said transmit path;

encapsulating said encrypted data packets into tunnel data frames by an encapsulation engine, within said transmit path; and

passing said tunnel data frames back to said local communications protocol stack for transmission to a physical network adapter on said remote server node.

18. The method of claim 17, wherein said transmit path further includes storing a digest value in a digest field in each of said tunnel data frames, said digest value equal to an output of a keyed hash function applied to said data packet encapsulated within said tunnel data frame concatenated with a counter value equal to a total number of tunnel data frames previously transmitted to said remote server node.

21. The method of claim 20, further comprising:

processing received data packets captured by said interface into said transport layer of said local communications protocol stack for capturing received data packets from said remote server node in a receive path;

decapsulating said received data packets by removing a tunnel frame header in an decapsulation engine, within said receive path; and

decrypting said received data packets in a decryption engine within said receive path;

passing said received data frames packets back to said local communications protocol stack for delivery to a user.